# Evaluating the FITTEST Automated Testing Tools: an Industrial Case Study

Cu D. Nguyen
Fondazione Bruno Kessler
Trento, Italy
Email: cunduy@fbk.eu

Bilha Mendelson, Daniel Citron, Onn Shehory
IBM Research Lab
Haifa, Israel
Email: {bilha, citron, onn}@il.ibm.com

Tanja E. J. Vos, Nelly Condori
Universidad Politécnica de Valencia
Valencia, Spain
Email: {tvos, nelly}@pros.upv.es

***Abstract*—This paper aims at evaluating a set of automated tools of the FITTEST EU project within an industrial case study. The case study was conducted at the IBM Research lab in Haifa, by a team responsible for building the testing environment for future development versions of an IBM system management product. The main function of that product is resource management in a networked environment. This case study has investigated whether current IBM Research testing practices could be improved or complemented by using some of the automated testing tools that were developed within the FITTEST EU project. Although the existing Test Suite from IBM Research ($TS_{ibm}$) that was selected for comparison is substantially smaller than the Test Suite generated by FITTEST ($TS_{fittest}$), the effectiveness of $TS_{fittest}$, measured by the injected faults coverage is significantly higher (50% vs 70%). With respect to efficiency, by normalizing the execution times, we found the $TS_{fittest}$ runs faster (9.18 vs. 6.99). This is due to the fact that the $TS_{fittest}$ includes shorter tests. Within IBM Research and for the testing of the target product in the simulated environment: the FITTEST tools can increase the effectiveness of the current practice and the test cases automatically generated by the FITTEST tools can help in more efficient identification of the source of the identified faults. Moreover, the FITTEST tools have shown the ability to automate testing within a real industry case.**

## I. Introduction

Software testing is the process of executing a program or system with the intent of finding defects [10]. It is currently the most important and widely used quality assurance technique applied in the industry. It may require over 50% of development budget and time [2]. Many test automation tools are currently available to aid test planning and control as well as test case execution and monitoring [5]. However, most of these tools, particularly those used in industrial practice, share a similar passive philosophy towards test case design, selection of concrete test data and test evaluation (i.e. oracles). They leave these crucial, time-consuming and demanding activities to the human testers. The lack of automation of these important testing activities means that the industry spends much effort and money on testing, nevertheless the quality of the resulting tests is sometimes low since they fail to find important errors in the system.

FITTEST, an EU funded research project, aims to overcome, at least to some extent, this problem. The FITTEST project involves a consortium of diverse competence, from execution monitoring, model-based testing [4], combinatorial testing [12], to search-based software engineering [7]. The ultimate goal of the project is to develop an Integrated Testing Environment (ITE for brevity) that consists of a suite of pluggable components that are being integrated for the FITTEST automated and continuous testing approach and tool set[15]. Being continuous, this testing approach will be suitable for testing fast evolving applications or those with dynamic and adaptive behaviors. i.e. the types of systems are envisaged to run on the Future Internet.

In this paper we present a case study for evaluating a subset of the FITTEST components, specifically those components that are responsible for Automated Test Case Design and Evaluation. The study has been executed at the IBM Research lab in Haifa[1] within a simulated test environment for an industrial system, called IT Management Product, or IMP for short. It is a large-scaled networked system that controls and optimizes resource management, such as creating and reconfiguring virtual machines on demand. Experimental evaluation of some of the techniques implemented in the FITTEST techniques have already been conducted and presented in earlier work [11]. In contrast, this study aims to obtain empirical evidence of the applicability of these techniques within the context of an industry team testing an industrial system. To this end we present a "which is better" type of case study [9]. These case studies are powerful since, although they cannot achieve the scientific rigor of formal experiments, the results of a case study can provide useful insights to help others judge whether the specific technology being evaluated could benefit their own organization. In order to assess tools, evaluative case study research must involve realistic systems and realistic subjects, as explicitly done in this study.

The contribution of this paper is twofold. On the one hand, it describes promising results of the use of automated test tool on an industrial case study. On the other hand, the study in this paper can serve as an example that others can follow when encountering the need to evaluate an automated testing tool.

The remainder of the paper is organized as follows: Section II describes the industrial context in which the study was performed. Section III presents the case study design framework, Finally, Section IV presents the results and Section V concludes the paper.

[1]This is a case study conducted by the Research team at IBM and does not necessarily reflect the development and the testing process of IBM products.

## II. Context- where was the case study performed

The study was executed at IBM Research Lab in Haifa. More specifically within the research team responsible for building the testing environment for future developments of an IT Management Product (IMP) (similar to [1]), a resource management system in a networked environment (more details in Section III-C1). At IBM Research Lab, the developers conduct limited amount of testing, the testing itself is conducted by this designated research team. It is working to enable the testing of new versions of the IMP by developing a simulated environment in which the system is executed.

The testing activities, described in this paper, have been done on IMP but in a simulated testing environment. The objective of the team was to identify whether current testing practices could be improved or complemented by using some of the new testing techniques that were introduced by the FITTEST EU project. For this purpose, the IBM Research team has used the FITTEST techniques and compared the results with the testing practices currently used during the initial steps of the Systems Verification Test (SVT). Only this level of tests was considered, since the next stage of the SVT testing is conducted elsewhere in IBM and so is beyond this case study.

Finally, since the IMP is a mature system, and in order to be able to measure fault-finding capability, several faults were injected into it within the simulated environment to mimic potential problems that had can be surfaced in such a system.

## III. Design of the case study

### A. Objective - What to achieve?

As indicated before, IBM Research wanted to evaluate the FITTEST automated testing tools to see if they are applicable to a selected System Under Test and how they compare to current practice. What makes a testing tool applicable in industry? First of all, it should be effective in finding faults! Second, this should be done efficiently, i.e. in a reasonable amount time. Finally, although finding faults in a reasonable amount of time is important, the amount of effort to set up and use the testing tools in the testing processes currently implanted should be important too. Hence, following [14], we focus on:

- ***RQ1*** [**Effectiveness**] Compared to the current test suite used for testing at IBM Research, can the FITTEST technologies contribute to the effectiveness of testing when it is used in the testing environments at IBM Research?
- ***RQ2*** [**Efficiency**] Compared to the current test suite used for testing at IBM Research, can the FITTEST technologies contribute to the efficiency of testing when it is used in the testing environments at IBM Research?
- ***RQ3*** [**Cost**] How much effort would be required to deploy the FITTEST technologies within the testing processes implanted at IBM Research?

### B. Cases or Treatments - What are being studied?

*1) Current test case design techniques used at IBM Research:* The STV test designed for the IT Management Product system used high level descriptions of complex customer use-cases to support exploratory test case design. The objective is to maximize the coverage of system use-cases.

*2) The FITTEST tools for automated test case design and evaluation:* The FITTEST testing approach is shown in Figure 1 and contains four phases:

1) *Logging* - Run the target application (SUT) and collect the logs it generates. This can be either real usage by end users of the application in the production environment, or test case execution in the test environment.
2) *Test-ware generation* - Analyse the logs to infer different testwares (i.e. FSM models, Oracles, Domain Input Specifications (DIS) and Test Cases).
3) *Test Execution* - The test cases are executed by running the SUT.
4) *Test evaluation* - The outcome of the running the test cases is evaluated using the oracles that are available.

For this case study, we instantiated two key components and their underpinning techniques of FITTEST. The two components are those responsible for Automated Test Case Design and Evaluation: **Logs2FSM** and **FSM2Tests**, that are done during the second phase (i.e. *Test-ware generation*):

- **Logs2FSM**, this component takes the logs generated by running the IMP system within the IBM Research simulation environment, and infers FSM models by applying the $k$-tail event-based model inference approach [3]. The *model-based oracles* that also result from this tool (see Figure 1) refer to the use of the paths generated from the inferred FSM as oracles. If these paths, when transformed to test cases, cannot be fully executed, then the tester needs to inspect the failing paths to see if that is due to some faults, or the paths themselves are infeasible.
- **FSM2Tests**, this component takes the output models of **Logs2FSM** and a Domain Input Specification (DIS) file created by a tester for the IBM Research SUT to generate concrete test cases. This component implements the *M*C* technique that combines model-based testing and combinatorial testing presented in [11]. This basically consist in: (1) generating test paths from the FSM (using algorithms that range from simple graph visit algorithms, to advanced techniques based on maximum diversity of the event frequencies, and semantic interactions between successive events in the sequence); (2) transform these paths into classification trees using the Classification Tree Editor (CTE XL)[2] [6] format, enriched with Domain Input Specifications (DIS) such as data types and partitions; (3) generate test combinations from those trees using t-way combinatorial criteria.

Since the system under test is an industrial one that is under development, no access is provided for external bodies. That is why IBM Research has to create a tool to transform the concrete test cases generated by **FSM2Tests** to an executable format (related to the "Automate Test Cases" oval in Figure 1).

[2]http://www.berner-mattner.com

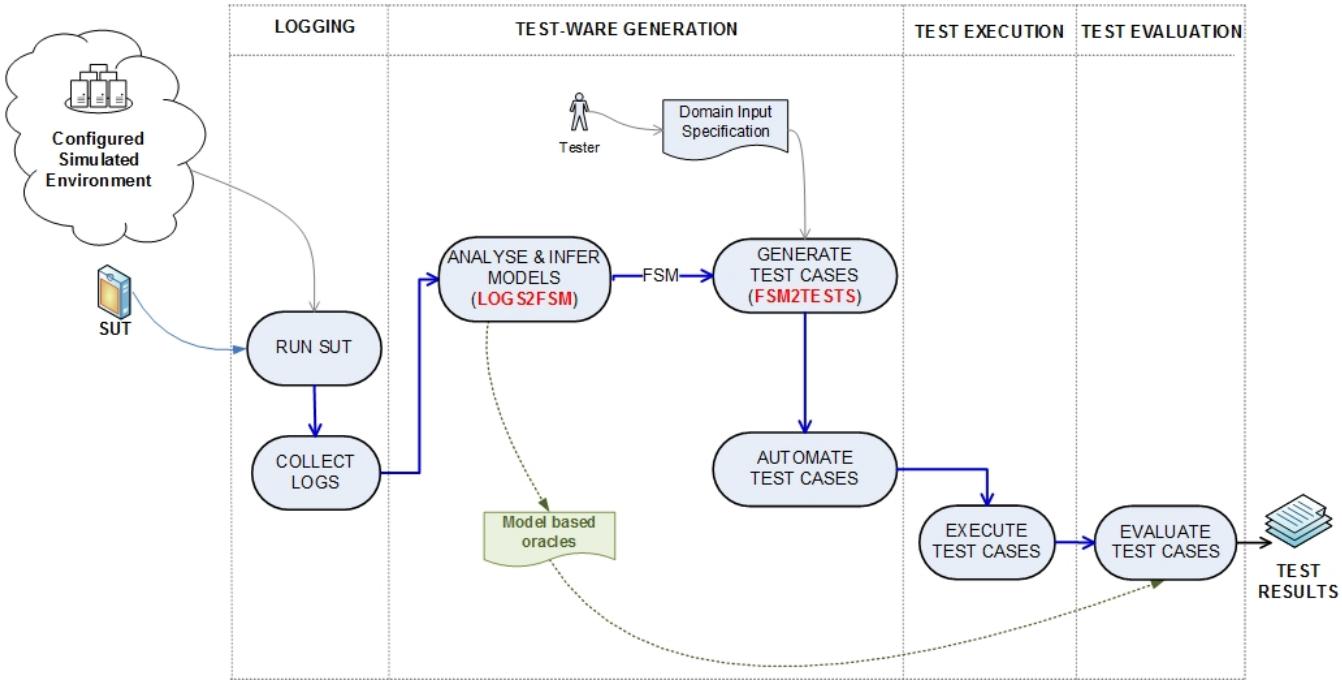


Fig. 1. The control flow of the FITTEST Automated Test Case Design and Evaluation. It contains 4 phases: *Logging*, *Test-ware generation*, *Test execution*, and *Test evaluation*.

Using this suite of tools, an input set of logs from IBM Reseach is transformed to a FITTEST test suite, namely $\text{TS}_{fittest}$.

### C. Objects of the study

*1) The System Under Test (SUT) :* The IBM IT Management Product (similar to [1]) is a distributed application for managing system resources in a networked environment. It consists of a Management Server (MS) that is an application that communicates with multiple managed clients and with users of the management system (typically system administrators). Managed clients are physical or virtual resources distributed over a network.

The IT Management Product system is used by IBM customers for managing IBM hardware and virtual devices, such as servers, Virtual Machines (VMs), switches and storage devices. The application has been developed for several years by IBM. IBM considers this application to be an important product, and hence is keen on assuring its quality through testing. IBM Research is developing the simulated environment to allow better testing of the system. Consequently, this simulated environment seems a good object for study.

The case study is to be performed on some new versions of this simulated system which are still under development and have not yet been released for customer use. Consequently, the case study system shares structure and protocols with the versions available to customers. As a result, public resources available for production version are relevant to the case study as well.

The term end-point is used in reference to the managed resources on client nodes, from the point-of-view of the management server. The MS executes operations on the end-points either per requests from the users, or autonomously based on defined policy (e.g. a recovery attempt after an end-point failure). The end-points have two-way communication with the MS. The communication operations supported are listed below (not all available for testing in this case study though): 1. Discovery: MS finds what end points are reachable; 2. Control: MS sends commands (configuration, OS restart) to end-points; 3. Query: MS queries status and configuration of and-points; 4. Report: End-points report problems to MS, and recovery from problems; and 5. Deploy: Deploy and obtain disposed of end-points.

The MS keeps an inventory of the current managed resources and their states in a standard database. The case system supports several protocols and message structures. Yet, in the case study, we are focusing only on HTTP messages. In the case system, these are the easiest to capture, interpret, and modify for case study purposes. Below we present an example of two messages, a request GET and a response in JSON format.

```
2013:02:28:17:20:08;GET
https://.../ibm.com:8422/ibm/.../rest/IMP/workloads/500165838;;200;{
   "workload" : {
      "hosts" : {
         "uri" : "https://.../workloads/500165838/hosts"
      },
      "createBy" : "root",
      "resilient" : "false",
      "approvalRequired" : "false",
      "oid" : "500165838",
      "state" : {
         "label" : "Started",
         "id" : 8
      },
      "virtualServers" : {
         "uri" : "https://.../workloads/500165838/virtualServers"
      },
      "specificationVersion" : "0.0",
      "changedDate" : 1362064498,
      "metrics" : {
         "uri" : "https://.../Server/500165838/.../monitordate"
      },
```

In this message, at time 2013:02:28:17:20:08 there was a

TABLE I. THE FAULTS INJECTED INTO THE SUT FOR EVALUATION

| Num | Injected Fault |
|---|---|
| IF1 | Fails to create unique new IDs |
| IF2 | Fails to open a JSON file |
| IF3 | Fails to validate workload size when resizing |
| IF4 | Fails to delete workload fails |
| IF5 | Fails to find a specific object ID |
| IF6 | Fails to create workload when resources are not enough for a deployment |
| IF7 | Fails to create a virtual server |
| IF8 | Fails to create a virtual disk |
| IF9 | Fails to get compatibilities |
| IF10 | Fails to write to a JSON file |

request to get the status of the workload with id 500165838. On successful execution this service returns an HTTP status code of 200 (OK) and the response in JSON format containing relevant information about the workload. A full list of the possible HTTP requests in the API of IBM IT Management Product can be found at [1].

*2) The existing Test Suite that was used for comparison with current practice ($TS_{ibm}$):* The IBM Research team had selected a set of the SVT tests as they were working on an initial testing stage. The SVT tests focuses on high-level and complex customer use-cases. It is done per release (2-3 times a year), and begins halfway through the release schedule. Most tests are manual, e.g. creating a configuration (a set of connected hardware devices), and running discovery by the management server. Some tests are automatic - automated GUI tests.

The tests that were chosen to form $TS_{ibm}$ are selected from the manual tests and were those tests that the team had identified as covering most of scenarios that are possible in the SUT. For the sake of this case study, for each test case in $TS_{ibm}$, an activation script has been written in order to execute each test case separately. Meaning, a test case is executed by calling the corresponding script and after each run the system is manually cleared to allow independence of the concrete tests execution.

*3) The faults that were injected:* IBM Research has injected 10 representative faults into the SUT to help better evaluating the different testing techniques. Those 10 injected faults were based on real faults that were identified in earlier time of the development and are listed in Table I.

The faults were manually injected in the code of the simulated environment. In order to distinguish them from other failures, we added a comment in the log files generated by IBM Research's simulation environment that started with "Injected bug". In this case study we will use these injected faults as a measurement to evaluate the fault-finding capability of the FITTEST tools being used.

### D. Subjects - Who apply the techniques?

The subjects are employees of IBM Research and FBK[3], a member of the FITTEST consortium.

The IBM Research subject was a 30-year old senior tester from IBM Research with 10 years of software development experience, 5 years of experience with testing of which 4 years with a system similar to the IMP and the approach described in Section III-B1. The tester had no previous knowledge of the FITTEST tools and holds a Computer Science degree.

The FBK subject was a researcher with more than 10 years of software development experience and more than 5 years of research experience in software testing and analysis.

[3]A research center located in Trento, Italy. URL: http://www.fbk.eu

### E. Variables - What are being measured?

The *independent variables* of the study setting are: The FITTEST testing techniques; the complexity of the industrial system; $TS_{ibm}$; the level of experience of testers of IBM Research that will use the techniques; the injected faults. The *dependent variables* are related those for measuring the applicability of the FITTEST tools in terms of effectiveness, efficiency and effort. Next we present their respective defined metrics:

1) Measuring effectiveness:
   a) amount of injected faults detected by both $TS_{ibm}$ and $TS_{fittest}$
   b) type of faults detected by both $TS_{ibm}$ and $TS_{fittest}$
2) Measuring efficiency. For both $TS_{ibm}$ and $TS_{fittest}$:
   a) size of the test suites: number of test cases and number of events (or commands)
   b) time needed to execute both the test suites

   Moreover for $TS_{fittest}$ we will measure:
   c) size of the original logs: number of events/-commands/requests
   d) metrics about the created FSM: number of nodes, number of transitions
3) Measuring effort in time (hours) for each of the subjects that was needed to create $TS_{ibm}$ and $TS_{fittest}$.

### F. Protocol

We adopted the following steps in this case study in order for the subjects to collaborate. At the same time, the protocol respects the business policies applied in IBM that allow little or no access to external parties.

1) Configure the simulated environment and create the logs [IBM Research subject]
2) Select test suite $TS_{ibm}$ [IBM Research subject]
3) Write activation scripts for each test case in $TS_{ibm}$ [IBM Research subject]
4) Generate $TS_{fittest}$ [FBK subject]
   a) Instantiate FITTEST components for the IMP
   b) Generate the FSM with Logs2FSM
   c) Define the Domain Input Specification (DIS)
   d) Generate the concrete test data with FSM2Tests
5) Select and inject the faults [IBM Research subject]
6) Develop a tool that transforms the concrete test cases generated by the FITTEST tool **FSM2Tests** to an executable format [IBM Research subject]
7) Execute $TS_{ibm}$ (run each activation script while manually clear the system after each activation script) [IBM Research subject]
8) Execute $TS_{fittest}$ [IBM Reseach subject]

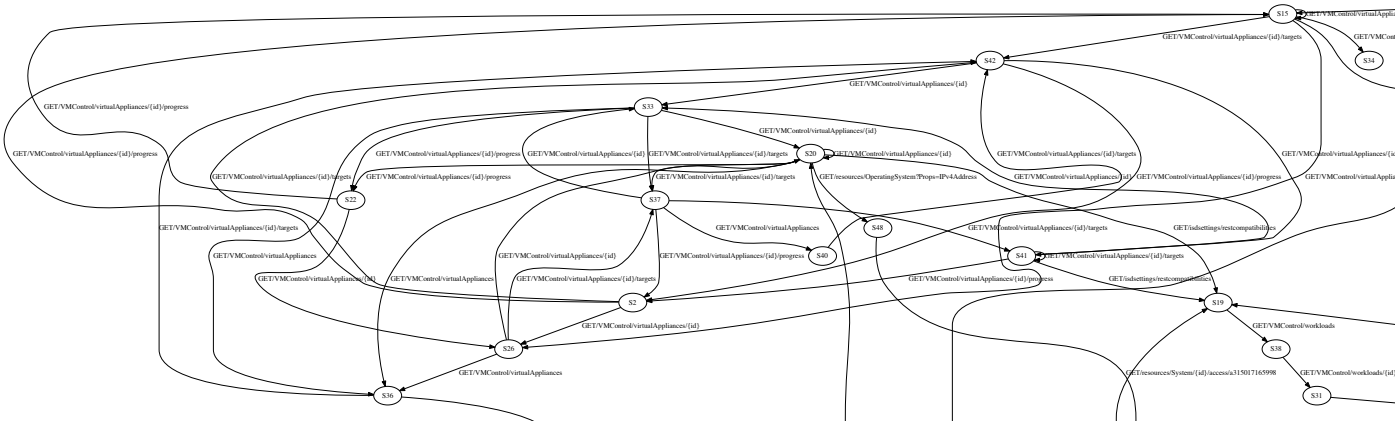

Fig. 2. A part of the model inferred for the case study. The model has 51 nodes and 134 transitions, inferred automatically. This figure illustrates how an inferred model looks like. In fact, it is machine generated and there is no need for a tester to read the model.

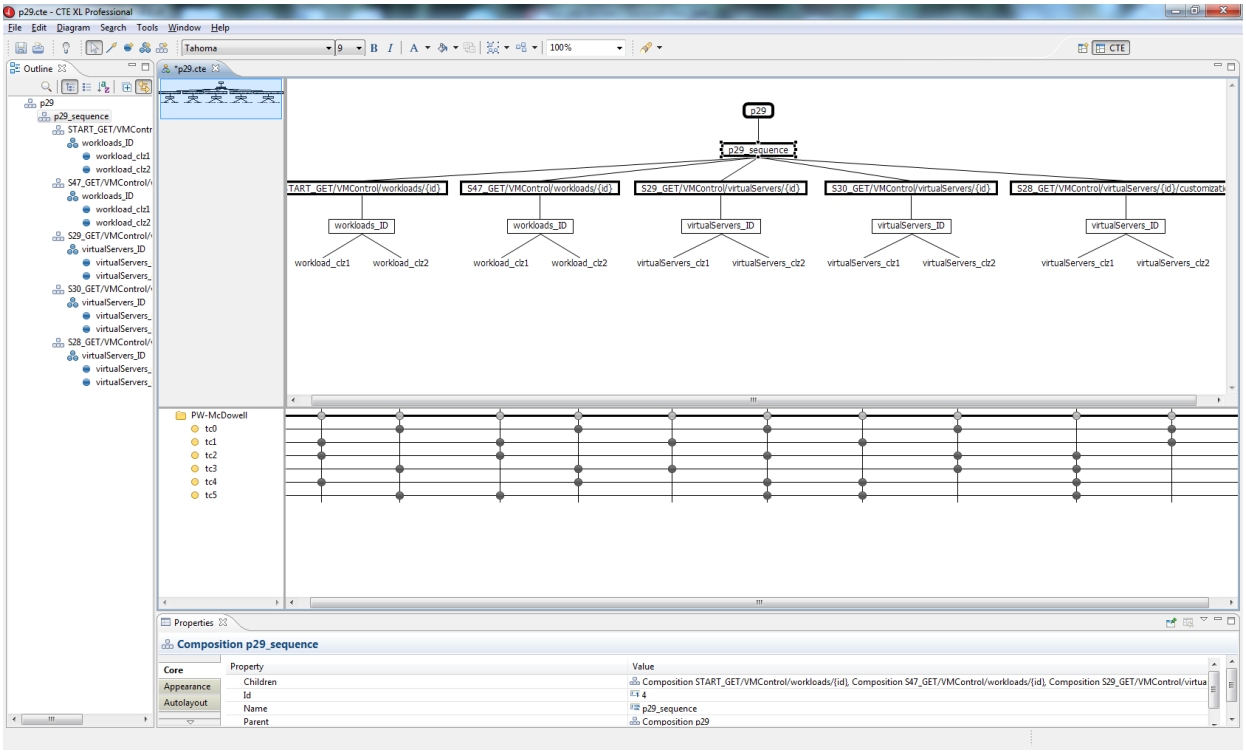


Fig. 3. An example of a test sequence and the test cases generated for the sequence. Each test sequence is transformed to a classification tree with input classifications. The combinations of input classes along the sequence are test cases that can be transformed to executable ones automatically.

## IV. Results and Discussion

### A. Results

This section summarizes and discusses the results and outcomes of this case study. We have followed the protocol presented in the previous section to measure the variables identified for the two test suites: $TS_{ibm}$ and $TS_{fittest}$. Table II contains the measures specific for the generated FSM model that was used to generate the FITTEST test suite $TS_{fittest}$. For illustrative reasons, part of this FSM can be found in Figure 2 and an example of a generated test sequence and corresponding test cases from the CTE XL[4] can be found in Figure 3. The figures give an idea how the models and test cases are visualized. They are generated and used automatically without human intervention. However, the tester can use graphical tools like CTE XL to enhance the models and tests, e.g. to put dependency on input data, if needed. In Table V the descriptive measures for both test suites are listed related to size and effort to create them. III contains the fault-finding capabilities of both test suites and Table IV the execution times. Since we only have one value for each test suite, no analysis techniques are available and the tables in this section just present the measured data on which the answers to the research questions from Section III-A are based.

**RQ1**: *Compared to the current test suite used for testing*

[4]http://www.berner-mattner.com/en/berner-mattner-home/products/cte

TABLE II. DESCRIPTIVE MEASURES FOR THE FSM THAT IS USED TO GENERATE $\mathbf{TS}_{fittest}$

| Variable | |
|---|---|
| Number of traces used to infer FSM | 6 |
| Average trace length | 100 |
| Number of nodes in generated FSM | 51 |
| Number of transitions in generated FSM | 134 |

*at IBM Research, can the FITTEST technologies contribute to the **effectiveness** of testing when it is used in the testing environments at IBM Research?*

TABLE III. EFFECTIVENESS MEASURES FOR BOTH TEST SUITES WITH RESPECT TO THE 10 INJECTED FAULTS. "0" MEANS THAT THE CORRESPONDING FAULT WAS NOT DETECTED, WHILE "1" MEANS IT HAS BEEN DETECTED.

| | IF1 | IF2 | IF3 | IF4 | IF5 | IF6 | IF7 | IF8 | IF9 | IF10 |
|---|---|---|---|---|---|---|---|---|---|---|
| TS_ibm | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| TS_fittest | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

As can be seen from Table V, the $\text{TS}_{ibm}$ is substantial smaller in size than the $\text{TS}_{fittest}$ in all parameters, this is one of the evident results of automation. However, not only the size of $\text{TS}_{fittest}$ is bigger, also the effectiveness of $\text{TS}_{fittest}$, measured by the injected faults coverage (see Table III), is significantly higher (50% vs 70%). Moreover, if we would combine the $\text{TS}_{ibm}$ and $\text{TS}_{fittest}$ suites, the effectiveness increases to 80%. Therefore, within the context of the studied environment, for IBM Research the FITTEST technologies can contribute to the effectiveness of testing and IBM Research has decided that, for optimizing faults-finding capability, the two techniques can best be combined.

**RQ2***: Compared to the current test suite used for testing at IBM Research, can the FITTEST technologies contribute to the **efficiency** of testing when it is used in the testing environments at IBM Research?*

TABLE IV. EFFICIENCY MEASURES FOR EXECUTION OF BOTH TEST SUITES.

| Variable | $\mathbf{TS}_{ibm}$ | normalized by size | $\mathbf{TS}_{fittest}$ | normalized by size |
|---|---|---|---|---|
| Execution Time with fault injection | 36.75 | 9.18 | 127.87 | 1.52 |
| Execution Time without fault injection | 27.97 | 6.99 | 50.72 | 0.60 |

It can be seen from Table IV that the time to execute $\text{TS}_{ibm}$ is smaller than the time to execute $\text{TS}_{fittest}$. This is due to the number of concrete tests in $\text{TS}_{fittest}$. When we normalize the execution time to the number of tests in the test suit, we see that per test, the $\text{TS}_{fittest}$ execution time is much smaller (1.52 vs. 9.18 minutes without the injected faults and 0.60 vs. 6.99 minutes with the injected faults). This is due to the fact that the $\text{TS}_{fittest}$ suite includes much shorter tests. The execution time is acceptable for IBM Research, considering the fact that the effectiveness of the tests can be improved and more faults can be detected in an efficient way (as was discussed in RQ1). Moreover, the shorter tests of which $\text{TS}_{fittest}$ is composed, can help identify the faults faster.

**RQ3***: How much effort would be required to deploy the FITTEST technologies within the testing processes implanted at IBM Research?*

As can be seen from Table V, the effort to set up the FITTEST components for the SUT and to specify the Domain Input Specification was 10 hours of effort for the FBK subject. Generating the FSM and the concrete test cases was automated by the tools. The whole CPU time needed was about 1 minute on a moderate personal computer. The effort to convert the concrete tests by the FITTEST tools into executable tests for IBM Research and writing the automated activation scripts was about 2.5 days for the experienced IBM Research subject.

The effort for the design and building the TSibm is less than half a day and is significantly lower than the time needed for the TSfittest (as described in Table IV). Part of the differences in the effort is due to the fact that TSibm was built using existing internal IBM testing tools.

In spite of the above the amount of effort needed to deploy and execute the FITTEST tools is found reasonable by IBM Research, considering the fact that these tasks need to be done only once during deployment. Moreover, the tools and format of the tests are new to the team, some learning is required. After all has been set-up, effort to generate a new FITTEST test suites when new logs would be available is fully automatic.

### B. Threats to validity

**Internal validity**. It is of concern when causal relations are examined. In our case study, an internal validity threat is related to the logs generated by the IBM simulation environment to be used for automatically constructing the test models. Because of the quality of models can be affected by the content of the input logs. We are aware of this threat and have asked IBM for a diverse set of logs. Another similar threat is that the quality of concrete test cases can be affected by the completeness of the Domain Input Specification (DIS) file because incomplete specification will weaken the efficiency of the $\text{TS}_{fittest}$. In fact, this threat might affect the overall number of detected faults by $\text{TS}_{fittest}$, but if the specification can be improved, such number can be greater. Therefore, the conclusion about the effectiveness of the $\text{TS}_{fittest}$ remains unchanged. Regarding to the involved subjects from IBM, although they had a high level of expertise and experience working in the industry as testers, they had no previous knowledge of the FITTEST tools. This threat was reduced by means of a closer collaboration between FBK and IBM, by complementing their competences in order to avoid possible mistakes or misunderstandings.

**External validity**. It is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case. Our results rely on one industrial case study using a given set of artificial faults. Although running such studies is expensive in terms of time consuming, we plan to replicate it with in order to have a more generalizable conclusions. However, as discussed earlier, the system under testing used is a typical of a broad category of industrial systems that communicates with multiple managed clients and with users of the management system.

**Construct validity**. This aspect of validity reflect to what extent the operational measures that are studied really represent what the researcher have in mind and what is investigated according to the research questions. This type of threat is

| | $\mathbf{TS}_{ibm}$ | | $\mathbf{TS}_{fittest}$ | |
|---|---|---|---|---|
| **size** | | | | |
| number of abstract test cases | NA | | 84 | |
| number of concrete test cases | 4 | | 3054 | |
| number of commands (or events) | 1814 | | 18520 | |
| **construction** | | | | |
| design of the test cases | manual cf. Section III-B1 | | automated cf. Section III-B2 | |
| **effort** | | | | |
| effort to create the test suite | design | 5 hours | set up FITTEST tools | 8 hours |
| | activation scripts | 4 hours | generate the FSM | automated, less than 1 second CPU time |
| | | | specify the DIS | 2 hours |
| | | | generate concrete tests | automated, less than 1 minute CPU time |
| | | | transform into executable format | 20 hours |

TABLE V. DESCRIPTIVE MEASURES FOR THE TEST SUITES $\mathbf{TS}_{ibm}$ AND $\mathbf{TS}_{fittest}$

mainly related to the use of injected faults to measure the fault-finding capability of our testing strategies. This is because the types of faults seeded may not be enough representative of real faults. In order to mitigate this threat, the IBM team identified representative faults that were based on real faults, identified in earlier time of the development. This identification although was realized by a senior tester, the list was revised by all IBM team that participated in this case study.

## V. CONCLUSIONS

We have presented a "which is better" [9] case study for evaluating FITTEST testing tools with a real user and real tasks within a realistic industrial environment of IBM Research. The design of the case study has been done according to the methodological framework for defining case studies presented in [13]. Although a one-subject case study will never provide general conclusions with statistical significance, the obtained results can be generalized to other similar software in similar testing environments of IBM Research [16], [8]. Moreover, the study was very useful for technology transfer purposes: some remarks during the study indicate that the FITTEST techniques would not have been evaluated in so much depth if it would not have been backed up by our case study design. Finally, having only limited number of subjects available, this study took several weeks to complete and hence we overcame the problem of getting too much information too late.

The objective of this research was to examine the advancements of the FITTEST tools and validate their potential to improve current testing practices at IBM Research. The following were the results of the case study:

- The FITTEST tools can increase the effectiveness of the current practice of the IBM Research team for testing the IMP within the simulated environment.
- The efficiency of the FITTEST tools is found acceptable by IBM Research for testing the IMP within the simulated environment.
- The test cases automatically generated by the FITTEST tools support better the identification of the source of the faults when testing the IMP within the simulated environment.
- The effort for deploying the FITTEST within a real industry case has been found reasonable by IBM Research.

Based on these results, IBM Research intends to discuss the FITTEST tools used in this case study with other parts of IBM to consider using similar approach in their testing processes.

Moreover, from the FITTEST project's point of view we have also learned: (i) the FITTEST tools have shown to be useful within the context of a real industrial case, and (ii) the FITTEST tools have the ability to automate the testing process within a real industrial case.

## ACKNOWLEDGMENT

This work was financed by the FITTEST project, ICT-2009.1.2 no 257574. Also, we would like to thank the great help we got from Alon Aradi for conducting the experiments.

## REFERENCES

[1] http://pic.dhe.ibm.com/infocenter/director/pubs/index.jsp?topic=

[2] B. Beizer. *Software Testing Techniques*. International Thomson Computer Press, 1990.

[3] A. Biermann and J. Feldman. On the synthesis of finite-state machines from samples of their behavior. *IEEE Trans. on Computers*, 21(6), 1972.

[4] A. C. Dias Neto, R. Subramanyan, M. Vieira, and G. H. Travassos. A survey on model-based testing approaches: a systematic review. In *Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*, WEASELTech '07, pages 31–36, New York, NY, USA, 2007. ACM.

[5] D. Graham and M. Fewster. *Experiences of Test Automation*. Pearson, 2012.

[6] M. Grochtmann and J. Wegener. Test case design using classification trees and the classification-tree editor cte. In *Proceedings of the 8th International Software Quality Week*, San Francisco, USA, Mai 1995.

[7] M. Harman and B. F. Jones. Search-based software engineering. *Information and Software Technology*, 43(14):833–839, 2001.

[8] W. Harrison. Editorial (N=1: an alternative for software engineering research). *Empirical Software Engineering*, 2(1):7–10, 1997.

[9] B. Kitchenham, L. Pickard, and S. Pfleeger. Case studies for method and tool evaluation. *Software, IEEE*, 12(4):52 –62, July 1995.

[10] G. J. Myers. *The Art of Software Testing*. John Wiley and Sons, 1979.

[11] C. D. Nguyen, A. Marchetto, and P. Tonella. Combining model-based and combinatorial testing for effective test case generation. In *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, pages 100–110. ACM, 2012.

[12] C. Nie and H. Leung. A survey of combinatorial testing. *ACM Comput. Surv.*, 43:11:1–11:29, February 2011.

[13] T. Vos, B. Marín, I. Panach, A. Baars, C. Ayala, and X. Franch. Evaluating software testing techniques and tools. In *Actas de XVI JISBD*, pages 531–536, 2011.

[14] T. E. J. Vos, B. Marín, M. J. Escalona, and A. Marchetto. A methodological framework for evaluating software testing techniques and tools. In *12th International Conference on Quality Software, Xi'an, China, August 27-29*, pages 230–239, 2012.

[15] T. E. J. Vos, P. Tonella, J. Wegener, M. Harman, W. Prasetya, and S. Ur. Testing of future internet applications running in the cloud. In S. Tilley and T. Parveen, editors, *Software Testing in the Cloud: Perspectives on an Emerging Discipline*, pages 305–321. 2013.

[16] A. Zendler, E. Horn, H. SchwŁrtzel, and E. Pldereder. Demonstrating the usage of single-case designs in experimental software engineering. *Information and Software Technology*, 43(12):681 – 691, 2001.